Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Previously presented) A method, comprising:
identifying a loop in a program;
identifying each vector memory reference in the loop;
determining dependencies between vector memory references in the loop, including determining unidirectional and circular dependencies;
distributing the vector memory references into a plurality of detail loops configured to allocate the vector memory references into a plurality of temporary arrays, sized and located, so that none of the vector memory references are cache synonyms, wherein the vector memory references that have circular dependencies therebetween are included in a common detail loop, and wherein the detail loops are ordered according to the unidirectional dependencies between the memory references;
analyzing an execution profile of the program after said distributing; and
based on the execution profile, determining whether to repeat said identifying a loop, said identifying each vector memory reference, said determining dependencies, and said distributing.

2. (Original) A method, as set forth in claim 1, further comprising allocating a plurality of temporary storage areas within a cache and determining the size of each temporary storage area based on the size of the cache and the number of temporary storage areas.

3. (Original) A method, as set forth in claim 1, further comprising at least one section loop including the plurality of detail loops.

4. (Original) A method, as set forth in claim 1, wherein distributing the vector memory references into a plurality of detail loops further comprises distributing the vector memory references into a plurality of detail loops that each contain at least one vector memory reference that could benefit from cache management.

5. (Original) A method, as set forth in claim 1, further comprising inserting cache management instructions into at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory.

6. (Original) A method, as set forth in claim 1, further comprising inserting prefetch instructions into at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory.

7. (Original) A method, as set forth in claim 1, further comprising performing loop unrolling on at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory.

8. (Original) A method, as set forth in claim 1, further comprising inserting at least one of a prefetch instruction and a cache management instruction into at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory, and performing loop unrolling on at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory.

9. (Previously presented) A method, comprising:
identifying a loop in a program;
identifying each vector memory reference in the loop;

determining dependencies between vector memory references in the loop; and

distributing the vector memory references into a plurality of detail loops that serially proceed through strips of the vector memory references and store the strips in temporary arrays so that none of the vector memory references are cache synonyms, wherein the vector memory references that have dependencies therebetween are included in a common detail loop;

wherein said distributing the vector memory references into a plurality of detail loops is performed by a first computer for execution by a second computer.

10. (Original) A method, as set forth in claim 9, further comprising allocating a plurality of temporary storage areas within a cache and determining the size of each temporary storage area based on the size of the cache and the number of temporary storage areas.

11. (Original) A method, as set forth in claim 9, further comprising at least one section loop including the plurality of detail loops.

12. (Original) A method, as set forth in claim 9, wherein distributing the vector memory references into a plurality of detail loops further comprises distributing the vector memory references into a plurality of detail loops that each contain at least one vector memory reference that could benefit from cache management.

13. (Original) A method, as set forth in claim 9, further comprising inserting cache management instructions into at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory.

14. (Original) A method, as set forth in claim 9, further comprising inserting prefetch instructions into at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory.

15. (Original) A method, as set forth in claim 9, further comprising performing loop unrolling on at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory.

16. (Original) A method, as set forth in claim 9, further comprising inserting at least one of a prefetch instruction and a cache management instruction into at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory, and performing loop unrolling on at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory.

17. (Previously presented) A method, comprising:
identifying a loop in a program;
identifying each vector memory reference in the loop;
determining dependencies between vector memory references in the loop;
distributing the vector memory references into a plurality of detail loops in response to cache behavior and the dependencies between the vector memory references in the loop, wherein the detail loops cause storage of the vector memory references in temporary arrays that are allocated consecutively so that no temporary arrays elements are cache synonyms,
wherein said identifying a loop, said identifying each vector memory reference, said determining dependencies between vector memory references and said distributing the vector memory references into

a plurality of detail loops produce code that is substantially independent of a computer architecture; and

performing code optimizations that are dependent on a computer architecture after said distributing.

18. (Original) A method, as set forth in claim 17, wherein distributing the vector memory references further comprises distributing the vector memory references into the plurality of detail loops with each loop having at least one of the identified vector memory references.

19. (Original) A method, as set forth in claim 17, further comprising determining dependencies between vector memory references in the loop, and wherein distributing the loop includes distributing the vector memory references into the plurality of detail loops, wherein the vector memory references that have circular dependencies therebetween are included in a common detail loop.

20. (Original) A method, as set forth in claim 17, further comprising inserting cache management instructions into at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory.

21. (Original) A method, as set forth in claim 17, further comprising inserting prefetch instructions into at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory.

22. (Original) A method, as set forth in claim 17, further comprising performing loop unrolling on at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory.

23. (Original) A method, as set forth in claim 17, further comprising inserting at least one of a prefetch instruction and a cache management instruction into at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory, and performing loop unrolling on at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory.

24. (Previously presented) A computer programmed to perform a method, comprising:

identifying a loop in a program;

identifying each vector memory reference in the loop;

determining dependencies between vector memory references in the loop; and

distributing the vector memory references into a plurality of detail loops configured to retrieve strips of the vector memory references and store the strips in temporary arrays, wherein the vector memory references that have circular dependencies therebetween are included in a common detail loop,

wherein the temporary arrays are configured to simultaneously fit in a single cache bank.

25. (Previously presented) A program storage medium encoded with instructions that, when executed by a computer, perform a method, comprising:

identifying a loop in a program;

identifying each vector memory reference in the loop;

determining dependencies between vector memory references in the loop; and

generating an expanded code of the program by distributing the vector memory references into a plurality of detail loops configured to allocate the vector memory references into temporary arrays that

avoid cache synonyms, wherein the vector memory references that have circular dependencies therebetween are included in a common detail loop,

wherein the expanded code is substantially independent of computer architectures.

26.-32. (Canceled).

33. (Previously presented) A method for reducing the likelihood of cache thrashing by software to be executed on a computer system having a cache, comprising:

executing the software on the computer system;

generating a profile indicating the manner in which the software uses the cache;

identifying a portion of the software that exhibits cache thrashing based on the profile data; and

modifying the identified portion of the software to reduce the likelihood of cache thrashing by distributing cache synonyms into detail loops configured to allocate the cache synonyms into temporary storage areas, sized and located, to prevent cache thrashing,

wherein said modifying occurs before optimizations that are based on an architecture of the computer system.

34. (Previously presented) A method, as set forth in claim 33, wherein modifying the identified portion of the software to reduce the likelihood of cache thrashing further comprises:

identifying a loop in the identified portion of the software;

identifying each vector memory reference in the identified loop;

determining dependencies between the vector memory references in the identified loop of the software, including determining unidirectional and circular dependencies; and

reducing cache thrashing by distributing the vector memory references into a plurality of detail loops, wherein the vector memory references that have circular dependencies therebetween are included in a common detail loop, and wherein the detail loops are ordered according to the unidirectional dependencies between the memory references.

35. (Previously presented) A method for reducing the likelihood of cache thrashing by software to be executed on a computer system having a cache, comprising:

executing the software on the computer system;

generating a profile indicating the manner in which the memory references of the software use the cache;

identifying a portion of the memory references based on the profile, wherein the portion of the memory references is determined to cause cache thrashing; and

reducing cache thrashing by distributing the portion of the memory references into distinct loops that allocate strips of the memory references into temporary arrays for execution,

wherein the temporary arrays are configured to simultaneously fit in a single cache bank.

36. (Previously presented) The computer of claim 24 wherein the temporary arrays are allocated consecutively such that no temporary array elements are cache synonyms.

37. (Previously presented) The computer of claim 24 wherein the details loops are allocated into section loops that cause iterative execution of the detail loops based on a size of the strips.

38. (Previously presented) The program storage medium of claim 25 wherein the temporary arrays are allocated consecutively such that no temporary array elements are cache synonyms.

39. (Previously presented) The method of claim 35 wherein the temporary arrays are located and sized to reduce cache thrashing.

40. (Previously presented) The method of claim 35 wherein the temporary arrays are allocated consecutively and iteratively executed by a size of the strips.